Data Structures in Java

Session 21

Instructor: Bert Huang

http://www.cs.columbia.edu/~bert/courses/3134

Announcements

- Homework 5 due
- Midterm solutions posted (sorry!)
- Homework 6 to be posted this weekend

Review

- Radix Sort specifics
- Comparison sorting algorithm characteristics
- Algorithms: Selection Sort, Insertion Sort, Shellsort, Heapsort, Mergesort, Quicksort

Today's Plan

- Sorting Algorithm Examples
- QuickSort space clarification
- External Sorting

Summary

	Worst Case Time	Average Time	Space	Stable?
Selection	$O(N^2)$	$O(N^2)$	$O(1)$	No
Insertion	$O(N^2)$	$O(N^2)$	$O(1)$	Yes
Shell	$O(N^{3/2})$	?	$O(1)$	No
Heap	$O(N \log N)$	$O(N \log N)$	$O(1)$	No
Merge	$O(N \log N)$	$O(N \log N)$	$O(N)/O(1)$	Yes/No
Quick	$O(N^2)$	$O(N \log N)$	$O(\log N)$	No

Selection Sort

3	7	5	2	6	1	0	4
0	7	5	2	6	1	3	4
0	1	5	2	6	7	3	4
0	1	2	5	6	7	3	4
0	1	2	3	6	7	5	4
0	1	2	3	4	7	5	6
0	1	2	3	4	5	7	6
0	1	2	3	4	5	6	7

Insertion Sort

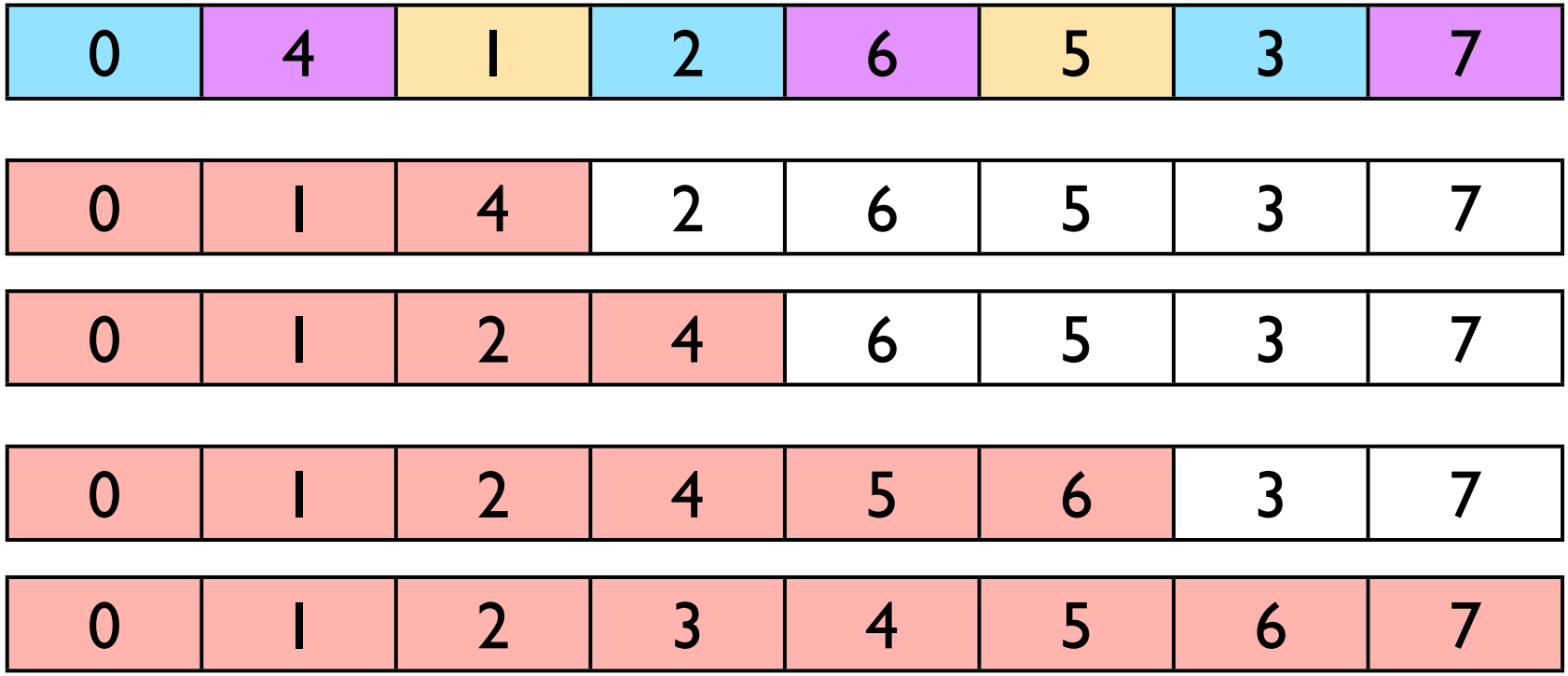
3	7	5	2	6	1	0	4
3	7	5	2	6	1	0	4
3	5	7	2	6	1	0	4
2	3	5	7	6	1	0	4
2	3	5	6	7	1	0	4
1	2	3	5	6	7	0	4
0	1	2	3	5	6	7	4
0	1	2	3	4	5	6	7

Shell Sort I

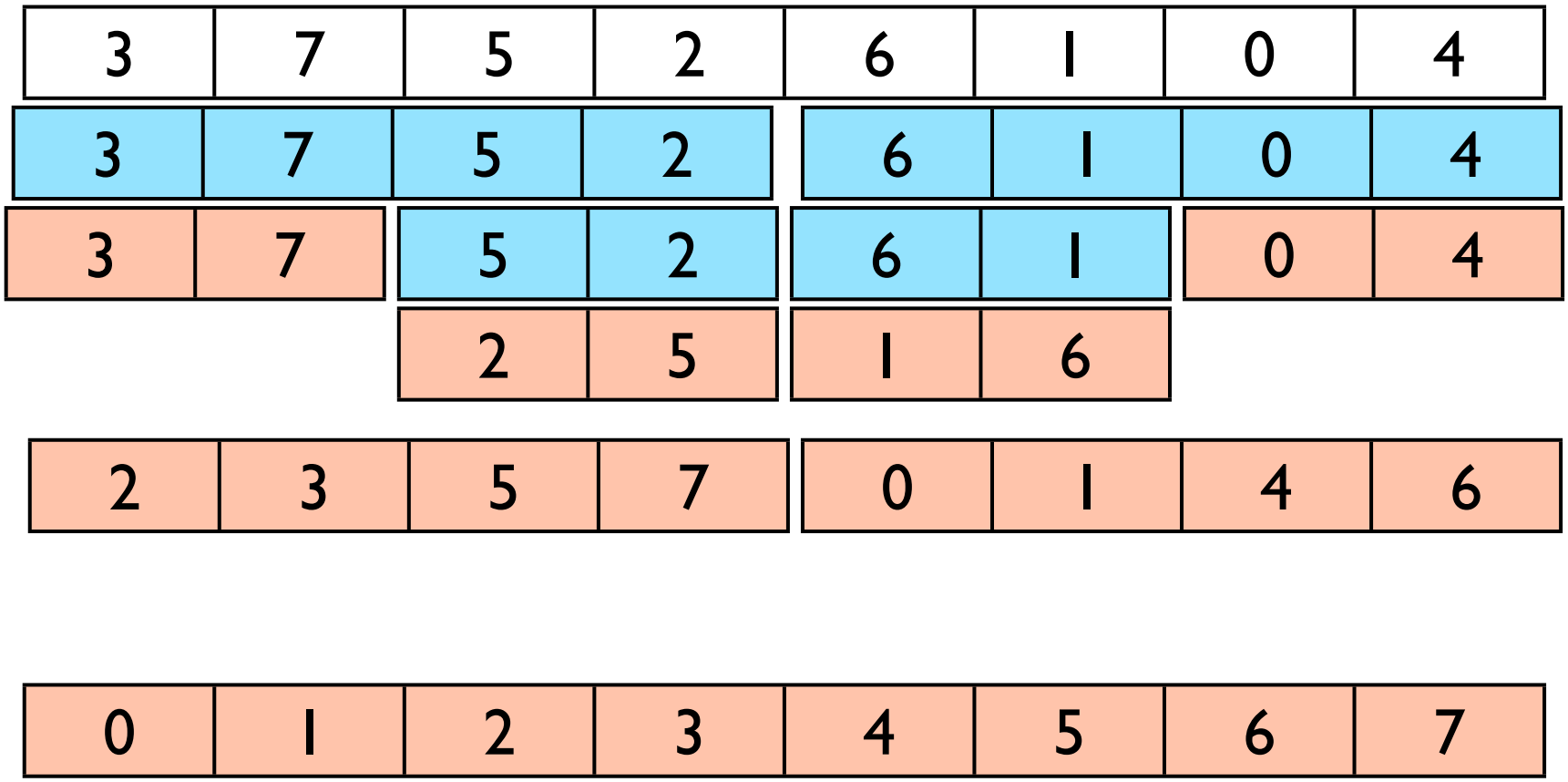
3	7	5	2	6	1	0	4
3	7	5	2	6	1	0	4
2	7	5	3	6	1	0	4
0	7	5	2	6	1	3	4
0	6	5	2	7	1	3	4
0	4	5	2	6	1	3	7

0	4	1	2	6	5	3	7

Shell Sort II



Merge Sort



Quick Sort

3	7	5	2	6	1	0	4
3	7	5	2	6	1	0	4
3	0	5	2	6	1	7	4
3	0	5	2	6	1	7	4
3	0	1	2	6	5	7	4
2	0	1	3	6	5	7	4
0	1	2	3	6	5	7	4
0	1	2	3	6	5	7	4
0	1	2	3	6	5	4	7
0	1	2	3	4	5	6	7

QuickSort Space

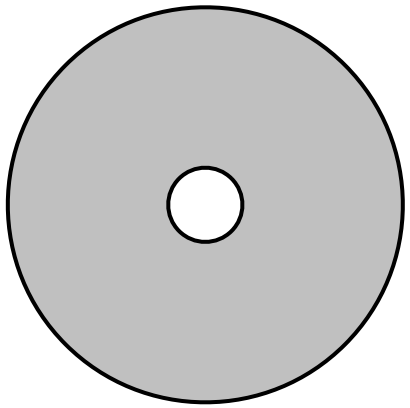
- QuickSort is a recursive algorithm
- Each recursive call sorts a segment of the array, it must store the beginning and end of the segment
- When the deepest recursive call is made, between N-1 and log N nested calls have occurred

External Sorting

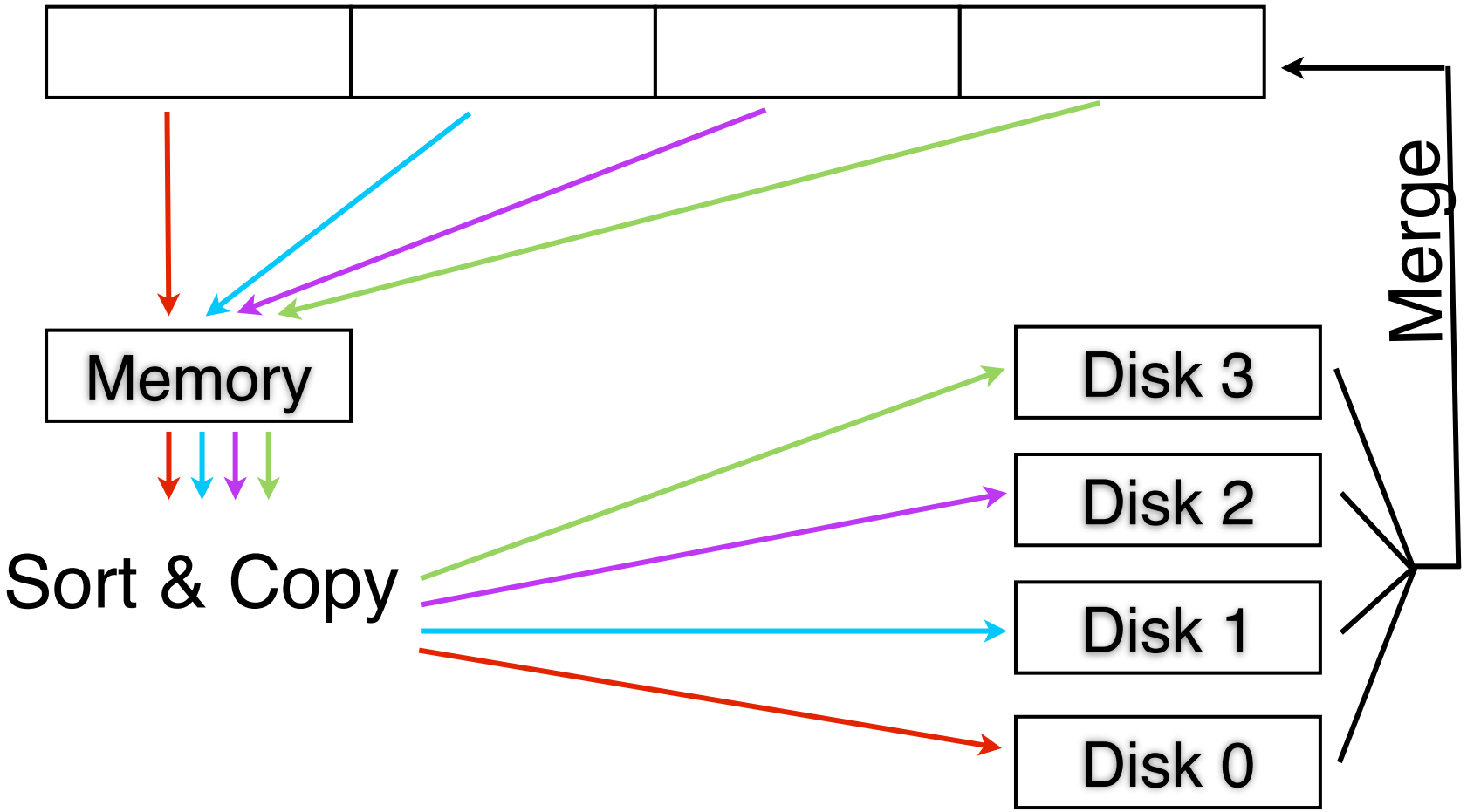
- So far, we have looked at sorting algorithms when the data is all available in RAM
- Often, the data we want to sort is so large, we can only fit a subset in RAM at any time
- We could run standard sorting algorithms, but then we would be swapping elements to and from disk
    - Instead, we want to minimize disk I/O, even if it means more CPU work

MergeSort

- We can speed up external sorting if we have two or more disks (with free space) via Mergesort
- One nice feature of Mergesort is the merging step can be done online with streaming data
- Read as much data as you can, sort, write to disk, repeat for all data, write output to alternating disks
  - merge outputs using 4 disks



4-way MergeSort



Simplified Running Time Analysis

- Suppose random disk i/o cost 10,000 ns
  - Sequential disk i/o cost 100 ns
  - RAM swaps/comparisons cost 10 ns
- Naive sorting: 10000 N log N
- Assume **M** elements fit in RAM.
  External mergesort:
  10 N log M + 100 N (# of sweeps through data)

Counting Merges

- After initial sorting, **N/M** sorted subsets distributed between 2 disks
- After each run, each pair is merged into a sorted subset twice as large.
  - Full data set is sorted after **log(N/M)** runs
- External sorting:
  10 N log M + 100 N log (N/M)
- (compare to 10000 N log N)

Reading

- http://www.sorting-algorithms.com/
- Weiss Chapter 7
  - Skim 7.4.1 (proof of Shell Sort)

- Enjoy break!